

characters from the text file within the indicated range. A stream of characters is then delivered to the browser 26.

DESCRIPTION AND DEFINITION TABLES

The names of pre-defined VRML fields are mapped to unique identifiers in a Field Descriptions table 136. Each row of the Field Descriptions table 136 stores values that represent a pre-defined VRML field. The Field Descriptions table 136 has columns named ID, Field Name, and Is Multi. The ID column stores a field identifier value that is uniquely associated with a field. The Field Name column stores the name of the field. The Is Multi column stores a Boolean value indicating whether the field is used to store multiple values. In one embodiment, the Field Descriptions table 136 stores the values set forth in Table 6 below.

TABLE 6

EXAMPLE VALUES OF FIELD DESCRIPTIONS TABLE		
ID	FIELD NAME	IS MULTI
1	SFBool	False
2	SFColor	False
3	MFCColor	True
4	SFFloat	False
5	MFFloat	True
6	SFImage	False
7	SFInt32	False
8	MFFInt32	True
9	SFNode	False
10	MFFNode	True
11	SFRotation	False
12	MFRotation	True
13	SFString	False
14	MFString	True
15	SFTime	False
16	MFTime	True
17	SFVec2f	False
18	MFVec2f	True
19	SFVec3f	False
20	MFVec3f	True

Thus, the purpose of the Field Descriptions table is to keep track of all defined field names and to a unique identifier. Values in the ID column are referenced by the node Script table 116 and Node Definition table 134.

A node Definitions table 134 provides a mapping of nodes to fields. Using the Node definitions table 134, given a particular node, all the fields of that node can be identified. Each row of the Node Definitions table 134 stores information that describes a field of a node. Thus each node defined with more than one field is represented by multiple rows of the Node Definitions table 134. The Node Definitions table 134 has columns named Node ID, field, ID, Field Name, and Description. The Node ID column stores a value that matches one of the identifiers stored in the ID column of the Node Descriptions table 124. The Field ID column store a value that matches one of the identifiers stored in the ID column of the Field Description table 136. The Field Name column stores the name of the field. The Description column stores an identifier that indicates the nature of the field, using the same values described above in connection with the Description column of the Script table 116.

In the preferred embodiment, values stored in the Node Definitions table 134 are derived from the VRML 2.0 language specification, which is published at <http://www.vrml.sgi.com/moving-worlds/>. Example values for the Anchor node of the VRML language are set forth in Table 7 below.

TABLE 7

EXAMPLE VALUES OF NODE DEFINITIONS TABLE			
NODE ID	FIELD ID	FIELD NAME	DESCRIPTION
Anchor	MFFNode	children	2
Anchor	SFString	description	2
Anchor	MFString	parameters	2
Anchor	MFString	url	2
10 Anchor	SFVec3f	bboxCenter	1
Anchor	SFVec3f	bboxSize	1
Anchor	MFFNode	addChildren	3
Anchor	MFFNode	removeChildren	3

For clarity, in Table 7 above the Node ID and Field ID columns show text values; however, in the preferred embodiment, as stated above, these columns store pointers, numeric values, or other non-textual identifiers that reference the Node Description table 124 and Field Description table 136.

PROCESSING VIRTUAL WORLDS USING A DATABASE

In one embodiment, the schema described above is implemented in a database system that is accessible to mechanisms for processing VRML worlds. In this embodiment, the mechanisms are implemented as one or more computer programs that can communicate with the database server 34 and exchange information with it including information that is stored in the database 20. Preferably, one of the computer programs provides the general functions listed in Table 8. It is convenient to refer to this program as the administration program. Alternatively, the administrative functions are implemented as administration pages 32 having a hyperlink associated with each function. Clicking on a hyperlink activates a Web server application that carries out the function.

TABLE 8

ADMINISTRATION PROGRAM FUNCTIONS	
1.	Import a virtual world into the database
2.	List all virtual worlds in the database
3.	Rename a virtual world in the database
4.	Delete a virtual world in the database
5.	Associate SQL statements with fields in the world

IMPORTING A VIRTUAL WORLD INTO A DATABASE

FIG. 3A is a flow diagram of a method of importing a virtual world into a database. In the preferred embodiment, the method shown in FIG. 3A is carried out using a VRML world as input and using a database organized according to the schema shown in FIG. 2A, 2B, and 2C.

In the embodiment shown in FIG. 3A, a method is initiated by logging into a database, such as the database server 34 that is managing tables organized in the schema shown in FIG. 2A, 2B, and 2C. Preferably, the administration pages 32 include a configuration mechanism whereby a system administrator can define the name of a database accessible to the VRML Agent and manager application. In step 302, a manager application presents a login dialog to the user that prompts the user to enter a user name, password, and database name. The user enters the requested information. If it matches the information provided by the system administrator in the configuration step, and the user name and password are recognized by the named database server, then a connection is established to the named database. In